

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : H04Q 3/00, G06F 17/00		A2	(11) International Publication Number: WO 96/21324
			(43) International Publication Date: 11 July 1996 (11.07.96)
(21) International Application Number: PCT/FI95/00719		(81) Designated States: AL, AM, AT, AU, AZ, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TT, UA, UG, US, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).	
(22) International Filing Date: 29 December 1995 (29.12.95)			
(30) Priority Data: 946209 30 December 1994 (30.12.94) FI			
(71) Applicant (for all designated States except US): NOKIA TELECOMMUNICATIONS OY [FI/FI]; Mäkkylän puistotie 1, FIN-02600 Espoo (FI).			
(72) Inventor; and (75) Inventor/Applicant (for US only): FINNI, Olli [FI/FI]; Roihuvuorentie 7 B 21, FIN-00820 Helsinki (FI).			
(74) Agent: OY KOLSTER AB; Iso Roobertinkatu 23, P.O. Box 148, FIN-00121 Helsinki (FI).			

Published

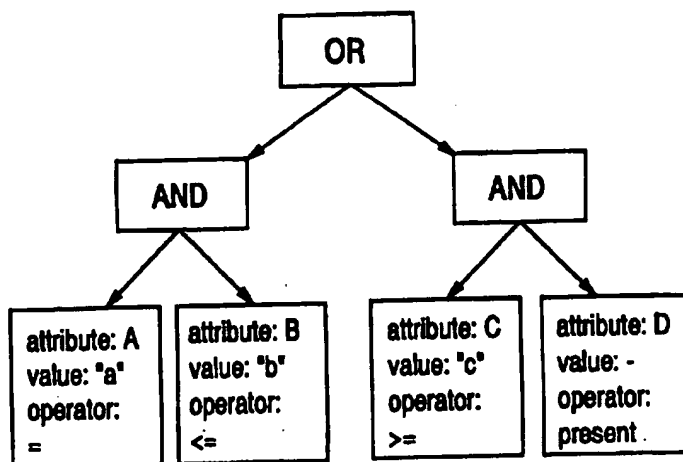
In English translation (filed in Finnish).

Without international search report and to be republished upon receipt of that report.

(54) Title: METHOD FOR COMPARING ATTRIBUTE VALUES OF CONTROLLABLE OBJECT EXPRESSIONS IN A NETWORK ELEMENT

(57) Abstract

The invention relates to a method for selecting a target group for an operation applied to a network element (NE) of a communication network. The method comprises the steps of (a) receiving from a network management element information on the operation and on the first target group (S), which is indicated as a group of object instances contained in the memory of the network element, and as comparison criteria of object instance attributes, the criteria containing the reference values, (b) comparing the attribute values of object instances belonging to the first target group to said reference values, and (c) selecting the final target group from among those object instances in the first target group (S) that have attributes fulfilling the received comparison criteria. To achieve a general-purpose method, a comparison between the attributes of a single object instance is carried out by (i) reading a semantic data which has been prestored in the network element memory, and which is associated with the attribute, (ii) searching the associated attribute value of the object instance from the network element, and (iii) comparing the attribute value to the received reference value by utilizing the semantic data read.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	GB	United Kingdom	MW	Malawi
AT	Austria	GE	Georgia	MX	Mexico
AU	Australia	GN	Guinea	NE	Niger
BB	Barbados	GR	Greece	NL	Netherlands
BE	Belgium	HU	Hungary	NO	Norway
BF	Burkina Faso	IE	Ireland	NZ	New Zealand
BG	Bulgaria	IT	Italy	PL	Poland
BJ	Benin	JP	Japan	PT	Portugal
BR	Brazil	KE	Kenya	RO	Romania
BY	Belarus	KG	Kyrgyzstan	RU	Russian Federation
CA	Canada	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic	KR	Republic of Korea	SE	Sweden
CG	Congo	KZ	Kazakhstan	SG	Singapore
CH	Switzerland	LI	Liechtenstein	SI	Slovenia
CI	Côte d'Ivoire	LK	Sri Lanka	SK	Slovakia
CM	Cameroon	LR	Liberia	SN	Senegal
CN	China	LT	Lithuania	SZ	Swaziland
CS	Czechoslovakia	LU	Luxembourg	TD	Chad
CZ	Czech Republic	LV	Latvia	TG	Togo
DE	Germany	MC	Monaco	TJ	Tajikistan
DK	Denmark	MD	Republic of Moldova	TT	Trinidad and Tobago
EE	Estonia	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	UG	Uganda
FI	Finland	MN	Mongolia	US	United States of America
FR	France	MR	Mauritania	UZ	Uzbekistan
GA	Gabon			VN	Viet Nam

Method for comparing attribute values of controllable
object expressions in a network element

5 The present invention in general relates to
management systems of communication networks. More
specifically, the invention concerns a method according to
the attached claim 1 for selecting a target group for an
operation applied to a network element of a communication
network.

10 The network management system referred to above may
in practise be of the type illustrated in figure 1.
Network operators, who are positioned at operation centers
OC, use network management workstations WS which are
connected to a separate workstation network WSN, such as,
15 for example, an Ethernet network. Typically, the
management system is distributed in several computers of
the workstation network, and they have an access to a
database DB which includes the information required for
the management of the network. Via a Q3 interface
20 determined in international standards, the management
system is connected, for instance, to a mobile network MN
which may comprise as network elements NE a mobile
services switching center MSC, a base station controller
BSC, a base station BTS and a mobile station. The
25 connection to the managed network is established by means
of a data communication network DCN. The managed network
may as well be any communication network, for example a
combined SDH/PDH network.

30 A CMISE application service element (CMISE = Common
Management Information Service Element), used in OSI
communication of Q3 interface in network management,
provides the application process with a SCOPE/FILTER
function (cf. ISO/IEC-9596-1 Information Technology - Open
Systems Interconnection - Common management information
35 protocol - Part 1:Specification). The purpose of the

function is to select a sub-group among the managed object instances, to which sub-group a CMIP operation (CMIP = Common Management Information Protocol), such as a m-Get, is applied.

5 The managed objects in the network element form a tree-like hierarchical structure, which is stored in the memory of the network element, and whose nodes contain the object instance and attributes relating thereto. In SCOPE/FILTER function, the SCOPE condition is used for
10 demarcating the CMIP operation to refer to a subtree in the tree (MIT, Managed Information Tree) of the managed object instances. This takes place by the name, i.e. DN (Distinguished Name), of the subtree root being given in the operation. To each object instance functioning as a
15 node for the subtree, the FILTER condition of the CMIP operation is applied separately. The values of the object instance attributes must fulfil this condition prior to applying the CMIP operation to the object instance. Following the application of SCOPE and FILTER conditions,
20 the group of those object instances to which the CMIP operation relates has been established.

 In testing whether the values of the object instances meet the FILTER condition, a comparison is carried out in which the attribute values of the object
25 instances are compared to specific reference values. Without a general-purpose testing mechanism, a separate, specific comparison function need to be created for each type of attribute in the network element. For carrying out the aforementioned, the network element must have a large
30 memory capacity. As far as large network elements are concerned, typically having hundreds or even thousands of managed objects, this complicates memory management and slows it down, because the overall memory capacity required for carrying out the comparison expands to large
35 measures. As regards small network elements, a memory

requirement such as this is also emphasized in economical sense, because in a strive for implementing small network elements (for example, mobile phones) as economically as possible, a memory expansion of just one byte bears significance.

For the above reasons, in the implementation of the FILTER condition, a general purpose solution should be the aim, i.e., how to implement the testing of the FILTER condition in a network element by a general-purpose manner. The object is to implement a function which is able to determine, on the basis of the FILTER condition of the CMIP operation and the attribute values of the object instance, whether the attribute values of the object instance fulfil the FILTER condition. The form of the FILTER condition is disclosed in the aforementioned standard ISO/IEC-9596-1, p. 16 by means of ASN.1 notation in an ASN.1 data type CMISFilter (ASN.1 = Abstract Syntax Notation One). (The ASN.1 notation is determined in the standard ISO 8824, in which a reader interested in the subject will find a more detailed description.) In practise, the FILTER condition is a regular expression (c.f. expressions in a programming language), which may contain logical operators (AND, OR, NOT), relational operators (=, <, >), graphic string operators (initialString, anyString, finalString) and set operators (present, subsetOf, supersetOf, non-nullIntersection).

The logical operators and the graphic string operators present no problem in testing a general-purpose FILTER condition. Instead, a problem arises concerning the implementation of relational operators and set operators. The difficulties in the implementation of the operators are clearly shown in the following example:

The attribute under comparison is assumed to be of type A, represented in ASN.1 notation as follows:

```
5      A ::= SEQUENCE {  
        B   INTEGER,  
        C   REAL,  
        D   GraphicString  
      }
```

The aforementioned denotation means that an attribute of type A is a sequence of three variables, B, C, and D, in which B is an integer, C is a real number and D is a graphic string. In the network element, the type A can be implemented as the following struct data type of C language:

```
15      typedef struct {  
        int      B;  
        float    C;  
        char     *D;  
      } A;
```

20 If the comparison function of the attribute values obtains the real value of the attribute and the reference value as a mere octet string, without knowing the structure of the data type in closer detail, it is hard to conceive how to carry out a general-purpose equivalence
25 comparison between the attribute values. Even if two values of the type A had exactly the same meaning, a linear octet string comparison results in an erroneous result if the graphic string pointers of the field D point to different locations. The equivalence comparison only
30 gives a correct result if the comparison function knows that "the data type A is a record whose third field (D) is a pointer to a graphic string, and that the equivalence comparison of the third field takes place between graphic strings addressed by the pointers".

35 Therefore, a problem in testing the FILTER condition stems from the fact that it is not possible to apply a linear octet string comparison to the attribute values contained in the FILTER condition, but the comparer must always, by one way or another, be aware of the data

types of the attribute values under comparison.

It is an object of the present invention to provide a method by which the aforementioned problem may be avoided. This object is achieved by a method according to the invention, characterized in that a comparison between the attributes of a single object instance is carried out by (a) reading a semantic data which has been prestored in the network element memory, and which is associated with the attribute, (b) searching the associated attribute value of the object instance from the network element, and (c) comparing the attribute value to the received reference value by utilizing the semantic data read.

The idea of the invention is to store in the memory of the network element (already at the commissioning stage, for example) the semantic data indicating the data type to be employed in comparing an attribute of a specific object instance. When the network element has been taken into use, and the network management system (operator) is searching for a target group for a CMIP management operation to be applied to the network element, this semantic data is first read at the network element, and then it is utilized in comparing the attribute values of the object instance to the reference values received from the network management system.

In the result according to the invention, it is assumed that the data types in accordance with the description language used (e.g. ASN.1) are created into data types of such a programming language for which there exists a translator (for example, C or C++ compiler). This takes place in the network element on the basis of unequivocal rules. In the above, an example is shown of ASN.1 data type and a data type in C language derived from the ASN.1 data type. The instances of data types (in C language, for example) derived from the ASN.1 representations are here referred to as home areas. In the

network element, the attribute values are thus represented as home areas.

5 The invention is based on the realization that attribute values can be compared, if the semantic of home areas corresponding to them is known in the network element when the CMIP management operation is being carried out.

10 The invention enables a considerably smaller memory and improves memory management in the network element. Although it is necessary to store a separate semantic data for each type of attribute in the network element, the significant point is that the semantic data can be stored in a considerably smaller memory space than comparison functions, which are required if no general-purpose
15 comparison mechanism according to the invention is available.

20 In the following, the invention and the preferred embodiments relating thereto will be disclosed in detail with reference to figures 2..5 in the attached drawings, in which

figure 1 shows a typical network management system,
figure 2 shows a tree formed by managed object instances,

25 figure 3 shows a semantic tree corresponding to data type A in C language, as described above,

figure 4 shows an example of a tree-like data structure corresponding to FILTER condition, and

30 figure 5 illustrates the operation of the invention in an exemplary situation in which the network operator searches a group of specific subscribers.

When a management operation is applied to a network element in a communication network, for which management operation a final target group must be found, the network management system sends, in accordance with the CMIP data
35 transfer protocol, information to the network element on

the operation and the target group. The target group is indicated as a group of object instances (i.e. a subtree, whose root node identifier is sent to the network element) and as comparison criteria of object instance attributes, which criteria contain one or more reference values and one or more comparison conditions. Following this, the attribute values of object instances belonging to the target group are compared in the network element to the received reference values by utilizing the received comparison conditions, and the final target group is selected from among those object instances in the original target group that have attributes fulfilling the received comparison criteria.

Figure 2 shows a tree of managed object instances, i.e. a MIT, and a group of object instances demarcated therefrom by SCOPE and FILTER conditions. The group of object instances (e.g. a subtree) demarcated by the SCOPE condition is within the broken line S. An arrow indicates the object instances demarcated by the FILTER condition "attribute A has the value 1". Thus, in the example of figure 2, the final target group of the CMIP operation is constituted by the object instances indicated by the arrow.

In the following, a more detailed description is given on how the comparison is carried out in the network element on the basis of information received from the network management system.

To achieve a general-purpose compare mechanism, the instances of data types, i.e. the home area semantic, is illustrated in the network element as a tree-like data structure, which may have nodes at several layers. Each node indicates the data type to be employed in comparing the attribute of the object instance. The tree-like data structure corresponds to a parsing tree that the translator (e.g. C compiler) of the programming language

used has produced out of the data type of the home area. It is advantageous to produce the semantic tree of the home area at the same time as the ASN.1 data type is being produced into data types of the programming language to be used in the network element. Typically, at this stage, a compiler program is utilized, which translates the ASN.1 representation into data types of the programming language.

A semantic tree shown in figure 3 corresponds to the C language version of the type A in the above example. By means of the semantic tree, the structure of the home area in the memory is known in detail. The semantic tree of the home area of the data type A indicates that the home area belongs to "struct" type of C language, and consists of two separate memory areas. The first memory area contains, in sequence, a four-byte-long integer (ASN.1 type INTEGER, C language type "int"), a four-byte-long floating point number (ASN.1 type REAL, C language type "float"), and the initial address of the second memory area (ASN.1 type GraphicString, C language type "char *", i.e. graphic string pointer). The second memory area contains the graphic string. Accordingly, by means of the semantic tree, it is known that "the equivalence comparison of the third field D of type A home areas takes place between graphic strings that are addressed by pointers positioned eight bytes from the beginning of the home areas". Correspondingly, the information contained in the semantic tree is utilized when comparing the first and the second fields of the home areas. Thus, by means of the semantic tree, it is possible to carry out a comparison between two attribute values of the type A as a comparison of the home areas corresponding to the attribute values.

It is assumed that the FILTER condition of a CMIP operation (e.g. m-Get) is represented in the network element (such as the Nokia DX200 switching exchange) as a

tree-like data structure whose leaf nodes include the Object Identifier of the attribute under comparison, the home area of the reference value and the operator. As the operator, there are the relational operator (\leq , \geq , $=$),
 5 the graphic string operator (initialString, anyString or finalString) or a string of set operators (present, subsetOf, supersetOf or nonNullIntersection). In the other nodes of the tree, there may be logical operators (AND, OR or NOT). Figure 4 shows an example of a tree-like data
 10 structure corresponding to the FILTER condition, in which the condition clause is ((attribute A=a) AND (attribute B \leq b)) OR ((attribute C \geq c) AND (attribute D is present)).

On the basis of the following algorithm, evaluate_filter, it is possible to produce a program code
 15 to test the FILTER condition. The recursive algorithm evaluate_filter goes through the tree-like data structure representing the FILTER condition as from the root, and returns the validity of the FILTER condition as the result. The algorithm tests the entire tree-like data
 20 structure except the leaf nodes. The test method for the conditions in the leaf nodes will be described in closer detail below. In the algorithm, the testing of the leaf nodes has been replaced by the operation compare_values. The algorithm does not take into consideration a situation
 25 in which the tree-like data structure might be erroneous as to its structure.

The algorithm evaluate_filter can be determined as follows:

```

30  boolean evaluate_filter (root node) {
    auxiliary variables: operator, child node, result;
        if (root node is empty)
            return TRUE;
        if (tree has root node only)
35         return compare_values (root node)
  
```

```
operator := operator included in root node;
child node := left son of root node;

if (operator = "NOT")
5   result :=not evaluate_filter (child node);
else
  do {
    result = evaluate filter (child node);
    if (child node is not empty)
10    child node := right-hand side brother of
        child node;
    } while (((result = FALSE and operator = "OR")
or (result = TRUE and operator = "AND")) and
        child node is empty);
15   return result;
}
```

20 The algorithm evaluate_filter indicates that the testing of logical operators in the FILTER condition presents no problems. The problem occurs in the testing of conditions in the leaf nodes of the tree-like data structure representing the FILTER condition, i.e. fulfilling the operation compare_values located in the evaluate_filter algorithm.

25 The method employed for comparing the attribute values contained in the FILTER condition takes place in two steps, as described above. According to the invention, the first step comprises constructing semantic trees and a directory for all the home areas of attribute values present in the network element. The semantic trees and the directory are stored into the network element. In the
30 second step, the algorithm compare_values, whose functioning is based on the interpretation of the stored semantic trees, is applied to the attribute values being
35 compared.

A prerequisite for the implementation of the algorithm `compare_values` is that the nodes of the semantic tree include the following information: the identifier of the ASN.1 type corresponding to the home area, the
5 identifier of the home area type, the offset for the fields of a record type home area from the beginning of the home area, and the size of the home area. In addition, there must exist a directory by means of which the Object Identifier registered to the attribute can be associated
10 with the semantic tree of the home area of the attribute. Figure 3 illustrates an example of a home area semantic tree.

The principle of the attribute value comparison is illustrated by the algorithm `compare_values`. The algorithm
15 is presented in a simplified form, and only equivalence comparison is treated in it. For example, the ASN.1 types SET and SET OF are not treated. A person skilled in the art is, however, able to formulate a complete algorithm on the basis of the description disclosed here. The leaf node
20 of the tree-like data structure of figure 4 representing the FILTER condition is supplied as an input to the algorithm `compare_values`. The leaf node contains the Object Identifier of the attribute employed in the comparison, the relational operator and the attribute
25 value employed in the comparison. In the algorithm, it is assumed that the attribute value corresponding to the attribute identifier can be retrieved when the comparison is started. As a result, the algorithm provides information on the success or failure of the attribute
30 value comparison. The algorithm does not take into account any possible error situations. The algorithm `compare_values` calls a recursive auxiliary algorithm `compare_home`.

35 Algorithm `compare_values`:

```
boolean compare_values (leaf node) {
  auxiliary variables: semantic tree_root,
                      attribute_value;
  semantic tree_root := search_semantic tree_root (
5      leafnode.attribute_identifier);
  attribute value := search_attribute_value (
                      leafnode.attribute_identifier);

  return compare_home (semantic tree_root,
10      attribute_value,
                      leaf node.attribute_reference
                      value);
}

15 Auxiliary algorithm compare_home:

boolean compare_home (root,
                      attribute_value,
                      attribute_reference value). {
20 auxiliary variables: child node, result

  if (root.home area_type = pointer type) {
    attribute_value := <interpreting attribute_value
25      as an address, and searching a new
      value from a location indicated by
      the address>;
    attribute_compare value := <interpreting
      attribute reference_value as an
      address, and searching a new value
30      from a location indicated by the
      address>;
  }
  if (root.ASN1 type = SEQUENCE) {
    child node := left son of root node;
35    do {
```

```

    result = compare_home (
        child node,
        attribute_value+child node.OFFSET,
        attribute_reference_value+child
5      node.
        OFFSET);
    child node := right brother of child node;
    while (result = TRUE and child node exists);
    return result;
10  } else {
    if (root.ASN1-type = BOOLEAN)
        return compare_boolean (
            attribute_value,
            attribute_reference_value);
15  else
    if (root.ASN1-type = INTEGER)
        return compare_integer (
            attribute_value,
            attribute_reference_value);
20  ...
    }
}
```

```

Auxiliary routine compare_boolean (compare_integer
25 equals):
```

```

boolean compare_boolean (value, reference value)
{
    return value = reference value;
30 }
```

As disclosed above, the comparison program of the network element reads the identity of the received attribute, reference value and the data type semantic. Following this, the comparison program requests the actual value of the attribute from a separate adaptation program

(which contains the information on where the attributes can actually be retrieved, and which is able to return the attribute value as the correct data type) and carries out the comparison. As a result, those object instances are
5 obtained whose attributes matched, and which the operation therefore is applied to.

To clarify the above, the following will examine a practical example with a telephone exchange as the network element. The network element contains a subscriber
10 register, storing information on the subscribers connected to the network element. At the network management interface, the subscriber is represented by the object class "Subscriber", having the attributes "Line number" and "Directory number". The example is imaginary, but the
15 attribute "Directory number" could refer, for example, to the number indicative of the subscriber line during signalling, and the "Line number" could be the identifier of an extension line within the network element. For the attribute "Line number", an ObjectIdentifier {1 2 3} is
20 registered, and for the attribute "Directory number" the identifier {1 2 4}. In ASN.1, the attributes are represented as follows:

Line number :: = INTEGER

Directory number :: = GraphicString

25 By means of an ASN.1 translator, the ASN.1 representations of the attributes can be translated into the following data type representations in C language:

typedef long Line number;

typedef char *Directory number;

30 It is assumed that the network management workstation (reference mark WS in figure 1) wants to find out the subscribers who are connected to the network element, and whose directory number begins with the digit 4 or 5. The network management workstation sends to the
35 network element the following CMIP operation m-Get, which

includes the first target group (S) and the FILTER condition, i.e. the comparison criteria of the attributes of the object instances. The group S is determined in the PDU (Protocol Data Unit) of the CMIP protocol, described
5 below by means of the ASN.1 notation, by the value of the baseManagedObjectInstance field, which indicates the subtree root node in the MIT of the network element, and by the scope value of the field. Fields irrelevant to the example have been left out and replaced by three dots. The
10 FILTER condition is indicated by the value of the filter field. As the root node, the example has an object instance representing the subscriber register. For reasons of clarity, the actual name of the subscriber register instance is in the example replaced by the identifier
15 <Subscriber register>. The value wholeSubtree of the scope field means that the group S contains all the object instances of the subtree, i.e. all the subscribers coupled to the network element. As the value of the filter field there is the ASN.1 expression, which may be described as
20 the expression "the directory number begins with the digit 4, or the directory number begins with the digit 5" in natural language.

```
{  
25     ...  
     baseManagedObjectInstance <Subscriber register>  
     ...  
     scope wholeSubtree,  
     filter or {  
30         item {  
             substrings {  
                 initialString {  
                     attributeId {1 2 4}  
                     string "4"  
35                 }  
             }  
         }  
     }  
}
```

```

    }
  }
  item {
    substrings {
5      initialString {
        attributeId {1 2 4}
        string "5"
      }
    }
10  }
  },
  ...
}

```

In the following, the numbers in parentheses refer
 15 to corresponding numbers in figure 5, which illustrates
 the operation of an agent in the network element. (The
 numbers, however, do not necessarily refer to the order of
 execution of the operations). In the network element, the
 m-Get operation is received (1) by an agent responsible
 20 for the execution of the operation in the network element.
 The agent organizes the FILTER condition of the m-Get
 operation into a tree-like configuration (2). The agent
 demarcates the group S, which is determined by the m-Get
 operation, from the object instance tree (i.e. MIT) of the
 25 network element (3). In figure 5, the group S is
 demarcated within the MIT by a broken line. The agent
 processes each object instance belonging to group S
 separately (4). The agent finds out whether a specific
 object instance belongs to the final group of object
 30 instances by calling the implementation of the algorithm
 evaluate_filter described above (5). On the basis of the
 attribute identifier {1 2 4} in the FILTER condition, the
 evaluate_filter retrieves the semantic tree corresponding
 to the home area of the attribute "Directory number" (6).
 35 The markings in the semantic tree in this exemplary case

mean that the attribute, whose identifier is {1 2 4}, is GraphicString as regards its ASN.1 type, and the data type of the home area is "char*" in C language, i.e. a graphic string pointer. Controlled by the information contained in the semantic tree, the evaluate_filter compares the comparison values included in the FILTER condition to the attribute "Directory number" of the object instance by a manner described above. Following this, as the result to the m-Get operation, the agent returns to the network management workstation those subscribers who have the directory numbers "56789" and "442224" (7). These subscribers constitute the final group of target instances. In the MIT of figure 5, an arrow points to the subscribers in question.

As noted above, the comparison method which is based on the utilization of semantic trees and used for comparing attribute values is a general-purpose one. The general-purpose feature here means that the method is independent of the ASN.1 representations of the attributes. In addition to the advantages described above, the general-purpose feature of the method achieves considerable savings concerning network element software development, because the changes in ASN.1 representations or the implementation of new ASN.1 representations will require no additional software development. For the above reason, the implementation of the method can be tested more thoroughly, which improves the reliability of the results of the comparison and of the entire network element.

It is obvious for a person skilled in the art that the various embodiments of the invention are not restricted to the examples above, but may vary within the scope of the attached claims.

Claims

1. A method for selecting a target group for an operation applied to a network element (NE) of a communication network, the method comprising the steps of
- 5 - receiving from a network management element information on the operation and on the first target group (S), which is indicated as a group of object instances contained in the memory of the network element, and as
- 10 comparison criteria of object instance attributes, the criteria containing reference values,
- comparing the attribute values of object instances belonging to the first target group to said reference values, and
- 15 - selecting the final target group from among those object instances in the first target group (S) that have attributes fulfilling the received comparison criteria,
- c h a r a c t e r i z e d in that
- the attribute of an object instance is compared
- 20 by
- reading a semantic data which has been prestored in the network element memory, and which is associated with the attribute,
- searching the associated attribute value of the
- 25 object instance from the network element, and
- comparing the attribute value to the received reference value by utilizing the semantic data read.
2. A method as claimed in claim 1, c h a r a c t e r i z e d in that the semantic data is
- 30 stored in the network element memory as a tree-like hierarchical structure.
3. A method as claimed in claim 2, c h a r a c t e r i z e d in that the semantic tree node contains at least (a) the identifier of the object
- 35 instance attribute, and (b) information on what data type

to employ in the comparison of said attribute value.

5

10

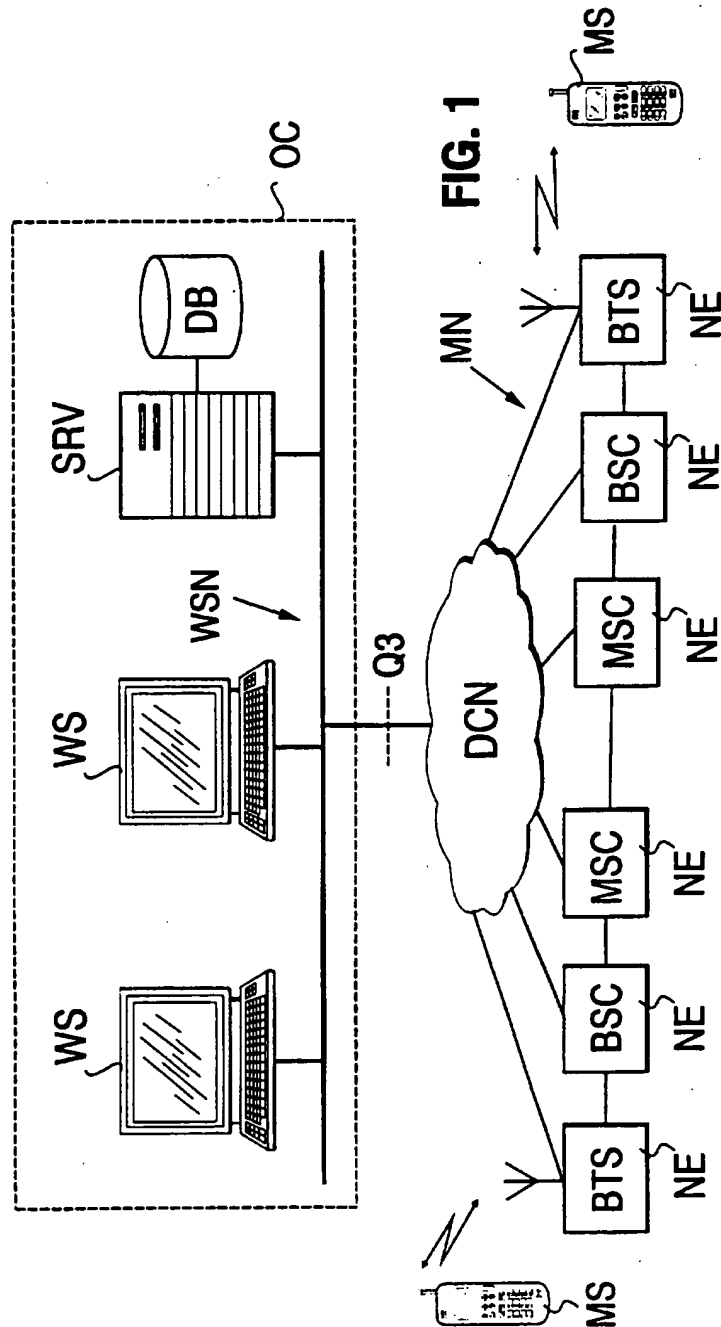
15

20

25

30

35



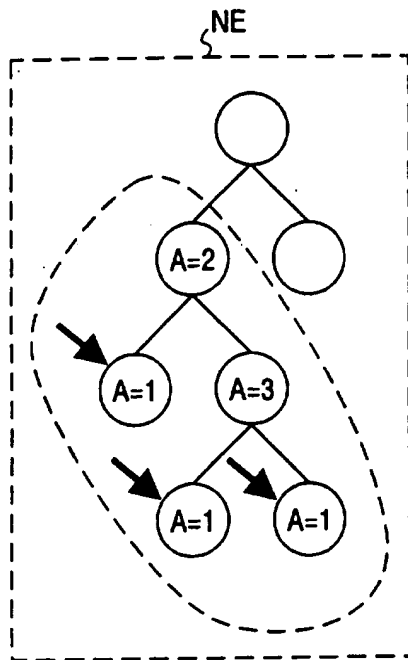


FIG. 2

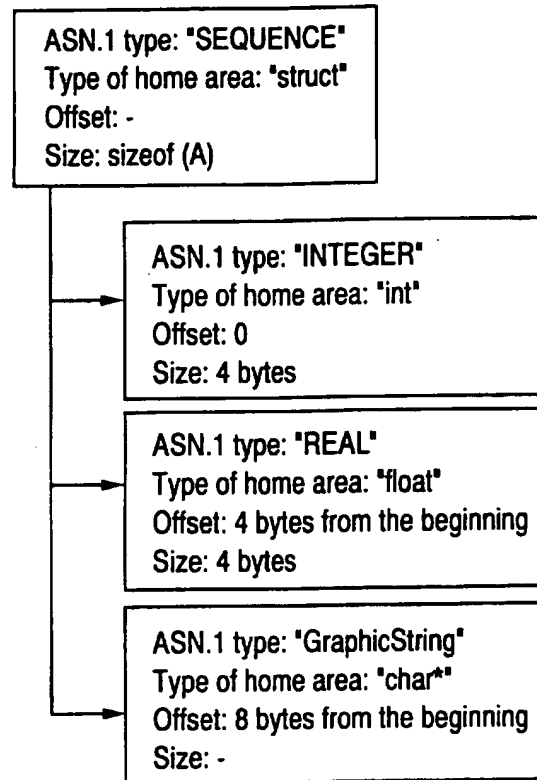


FIG. 3

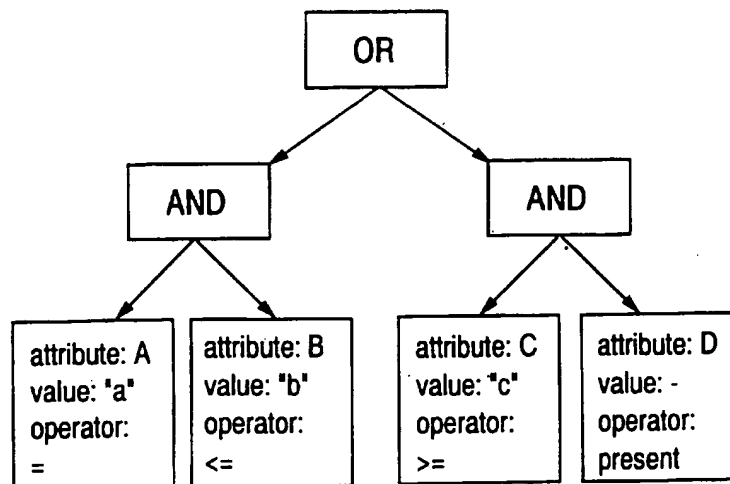


FIG. 4

3/3

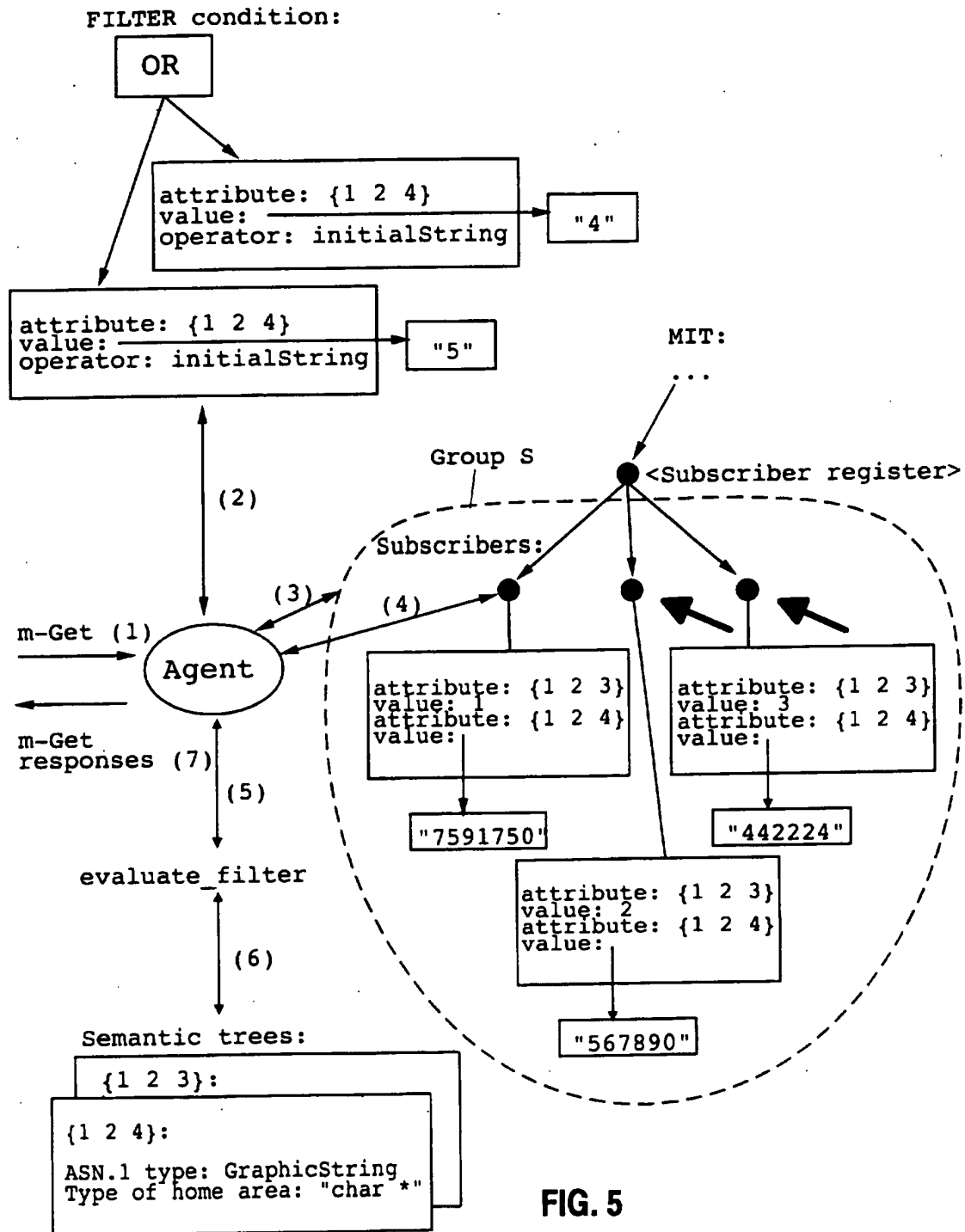


FIG. 5